UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/698,820	10/31/2003	Matthew Englehart	MWS-062RCE2	1288

74321   7590   01/21/2009
LAHIVE & COCKFIELD, LLP/THE MATHWORKS
FLOOR 30, SUITE 3000
One Post Office Square
Boston, MA 02109-2127

EXAMINER
CHEN, QING

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
01/21/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 November 2008*.
2a)☐ This action is **FINAL**.   2b)☒ This action is non-final.
3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-26* is/are pending in the application.
    4a) Of the above claim(s) _____ is/are withdrawn from consideration.
5)☐ Claim(s) _____ is/are allowed.
6)☒ Claim(s) *1-26* is/are rejected.
7)☐ Claim(s) _____ is/are objected to.
8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.
10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.
    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
    a)☐ All b)☐ Some * c)☐ None of:
        1.☐ Certified copies of the priority documents have been received.
        2.☐ Certified copies of the priority documents have been received in Application No. _____.
        3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
    Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413)
    Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1.   This Office action is in response to the amendment filed on November 10, 2008, entered

by the RCE filed on the same date.

2.   **Claims 1-26** are pending.

3.   **Claims 1 and 16** have been amended.

4.   The objections to Claims 1-26 are withdrawn in view of Applicant's amendments to the

claims.

5.   It is noted that Claim 19 contains no proposed amendments. However, the claim bears the

"currently amended" status identifier.

Continued Examination Under 37 CFR 1.114

6.   A request for continued examination under 37 CFR 1.114, including the fee set forth in

37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible

for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been

timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR

1.114. Applicant's submission filed on November 10, 2008 has been entered.

Response to Amendment

Claim Rejections - 35 USC § 103

7.   The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in
> section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are
> such that the subject matter as a whole would have been obvious at the time the invention was made to a person

having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8.   **Claims 1-8, 11-23, and 26** are rejected under 35 U.S.C. 103(a) as being unpatentable over "**Real-Time Workshop® User's Guide," January 1999 (hereinafter "RTW_UG")** in view of **US 2003/0056195 (hereinafter "Hunt")**.

        As per **Claim 1**, RTW_UG discloses:

-      providing an automatic code generator to create source code that implements functionality of said at least one graphical model and that corresponds to data referenced by said at least one graphical model *(see Figure 3-1; Chapter 3-3, "The Real-Time Workshop automates the task of building a stand-alone program from your Simulink model.")*; and

-      specifying a first manner in which said automatic code generator creates said source code *(see Chapter 3-6, "All pages of the Simulation Parameters dialog box affect Real-Time Workshop code generation.")*.

        However, RTW_UG does not disclose:

-      providing a user interface with a plurality of selectable parameters; and

-      creating a custom storage class in said graphical modeling and execution environment utilizing parameters selected by a user from said plurality of selectable parameters, said custom storage class specifying a second manner in which said automatic code generator creates source code corresponding to said data referenced by said at least one graphical model in said graphical modeling and execution environment, said second manner differing from said first manner.

        Hunt discloses:

- providing a user interface with a plurality of selectable parameters *(see Figure 2;*

Paragraph [0049], "The code generator tool of the preferred embodiment of the present

invention is embodied as an application program that presents the user with a Graphical User

Interface (GUI) that can be used to easily input meta data about an object model."); and

- creating a custom storage class in said graphical modeling and execution environment

utilizing parameters selected by a user from said plurality of selectable parameters, said custom

storage class specifying a second manner in which said automatic code generator creates source

code corresponding to said data referenced by said at least one graphical model in said graphical

modeling and execution environment, said second manner differing from said first manner *(see*

Paragraph [0050], "Turning now to FIG. 1, by way of an overview, the user of the code

generator 100 ... of the present invention enters meta-data 102 into the code generator 100's

GUI."; Paragraph [0083], "Consider now the entry of the meta data used in connection with the

present invention. Such information representing the meta data is input via the GUI such as

illustrated in FIGS. 2-6."; Paragraph [0088], "When the user is ready to build the class, the

"Generate" button 218 can be selected, for example by clicking with a mouse.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to include

providing a user interface with a plurality of selectable parameters; and creating a custom storage

class in said graphical modeling and execution environment utilizing parameters selected by a

user from said plurality of selectable parameters, said custom storage class specifying a second

manner in which said automatic code generator creates source code corresponding to said data

referenced by said at least one graphical model in said graphical modeling and execution

environment, said second manner differing from said first manner. The modification would be

obvious because one of ordinary skill in the art would be motivated to apply the same user edits

to model data without introducing errors *(see Hunt – Paragraph [0004])*.

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and RTW_UG further

discloses:

- providing a view of salient aspects of said source code generated by said automatic

code generator utilizing the user-selected parameters *(see Chapter 11-6, "MdlStart Function*

Custom Code dialog box").

As per **Claim 3**, the rejection of **Claim 2** is incorporated; however, RTW_UG does not

disclose:

- changing said user-selected parameters for said custom storage class in said user

interface; and

- adjusting said source code generated by said automatic code generator to reflect said

change in said user-selected parameters.

Hunt discloses:

- changing said user-selected parameters for said custom storage class in said user

interface *(see Paragraph [0050], "When a change to the object model needs to be made, the user*

can simply update the meta data through the GUI and regenerate all the files associated with the

object model."); and

- adjusting said source code generated by said automatic code generator to reflect said change in said user-selected parameters *(see Paragraph [0050], "When a change to the object model needs to be made, the user can simply update the meta data through the GUI and regenerate all the files associated with the object model.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of <u>Hunt</u> into the teaching of <u>RTW_UG</u> to include changing said user-selected parameters for said custom storage class in said user interface; and adjusting said source code generated by said automatic code generator to reflect said change in said user-selected parameters. The modification would be obvious because one of ordinary skill in the art would be motivated to apply the same user edits to model data without introducing errors *(see <u>Hunt</u> – Paragraph [0004])*.

As per **Claim 4**, the rejection of **Claim 3** is incorporated; and <u>RTW_UG</u> further discloses:

- displaying salient aspects of said adjusted source code in said view of salient aspects of said source code *(see Chapter 11-6, "MdlStart Function Custom Code dialog box")*.

As per **Claim 5**, the rejection of **Claim 2** is incorporated; and <u>RTW_UG</u> further discloses:

- wherein said view of salient aspects of said source code automatically generated includes at least one token, said at least one token being symbolically representative of a non-displayed segment of said source code *(see Chapter 3-23, "Template makefiles are essentially*

makefiles in which certain tokens are expanded to create a makefile for your model (model.mk).

model.mk is created from the template makefile specified in the Real-Time Workshop page of the

Simulation Parameters dialog box by copying each line from the template makefile and

expanding the tokens of Table 3-4.").

As per **Claim 6**, the rejection of **Claim 1** is incorporated; however, RTW_UG does not

disclose:

- wherein said custom storage class declares macros for instances of constant data.

Hunt discloses:

- wherein said custom storage class declares macros for instances of constant data *(see*

Paragraph [0050], "This meta-data represents a hierarchy of interfaces that include attributes

and operations and is stored in a SQL compliant database in the preferred embodiment. From

this data, the current embodiment of the code generator generates the following seven C++ files

plus four database SQL files where applicable: ...").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to include

wherein said custom storage class declares macros for instances of constant data. The

modification would be obvious because one of ordinary skill in the art would be motivated to

apply the same user edits to model data without introducing errors *(see Hunt – Paragraph*

[0004]).

As per **Claim 7**, the rejection of **Claim 1** is incorporated; however, RTW_UG does not disclose:

- wherein said custom storage class declares variables for instances of constant data.

Hunt discloses:

- wherein said custom storage class declares variables for instances of constant data

(see Paragraph [0050], "This meta-data represents a hierarchy of interfaces that include attributes and operations and is stored in a SQL compliant database in the preferred embodiment. From this data, the current embodiment of the code generator generates the following seven C++ files plus four database SQL files where applicable: ...").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to include wherein said custom storage class declares variables for instances of constant data. The modification would be obvious because one of ordinary skill in the art would be motivated to apply the same user edits to model data without introducing errors *(see Hunt – Paragraph [0004]).*

As per **Claim 8**, the rejection of **Claim 1** is incorporated; however, RTW_UG does not disclose:

- wherein said user-selected parameters control at least one of a manner in which automatically generated source code is defined, declared, accessed and addressed.

Hunt discloses:

- wherein said user-selected parameters control at least one of a manner in which automatically generated source code is defined, declared, accessed and addressed *(see Figure 2; Paragraph [0049], "The code generator tool of the preferred embodiment of the present invention is embodied as an application program that presents the user with a Graphical User Interface (GUI) that can be used to easily input meta data about an object model."; Paragraph [0050], "Turning now to FIG. 1, by way of an overview, the user of the code generator 100 ... of the present invention enters meta-data 102 into the code generator 100's GUI."; Paragraph [0083], "Consider now the entry of the meta data used in connection with the present invention. Such information representing the meta data is input via the GUI such as illustrated in FIGS. 2-6.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of <u>Hunt</u> into the teaching of <u>RTW_UG</u> to include wherein said user-selected parameters control at least one of a manner in which automatically generated source code is defined, declared, accessed and addressed. The modification would be obvious because one of ordinary skill in the art would be motivated to apply the same user edits to model data without introducing errors *(see <u>Hunt</u> – Paragraph [0004]).*

As per **Claim 11**, the rejection of **Claim 1** is incorporated; however, <u>RTW_UG</u> does not disclose:

- creating a separate header file with said automatic code generator in response to said selection of one of said plurality of user-selected parameters.

<u>Hunt</u> discloses:

- creating a separate header file with said automatic code generator in response to said

selection of one of said plurality of user-selected parameters *(see Paragraph [0051], "1. A*

hierarchy of C++ Interface definition files shown as interface.h header files 106 ...").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate the teaching of <u>Hunt</u> into the teaching of <u>RTW_UG</u> to include

creating a separate header file with said automatic code generator in response to said selection of

one of said plurality of user-selected parameters. The modification would be obvious because

one of ordinary skill in the art would be motivated to apply the same user edits to model data

without introducing errors *(see <u>Hunt</u> – Paragraph [0004]).*

As per **Claim 12**, <u>RTW_UG</u> discloses:

- a processor for:

    - executing an automatic code generator to create source code that implements

functionality of said at least one graphical model and that corresponds to data referenced by said

at least one graphical model *(see Figure 3-1; Chapter 3-3, "The Real-Time Workshop automates*

the task of building a stand-alone program from your Simulink model."); and

    - specifying a first manner in which said automatic code generator creates said

source code *(see Chapter 3-6, "All pages of the Simulation Parameters dialog box affect Real-*

Time Workshop code generation."); and

- a display device for:

- displaying a view of salient aspects of said source code generated by said automatic code generator utilizing said user-selected parameters *(see Section 11-6, "MdlStart Function Custom Code dialog box")*.

However, <u>RTW_UG</u> does not disclose:

- creating a custom storage class in said modeling and execution environment, said custom storage class created utilizing parameters selected by a user from a plurality of selectable parameters; and

- displaying a user interface with said plurality of selectable parameters for said custom storage class, said custom storage class specifying a second manner in which said automatic code generator creates source code corresponding to said data referenced by said at least one graphical model, said second manner differing from said first manner.

<u>Hunt</u> discloses:

- creating a custom storage class in said modeling and execution environment, said custom storage class created utilizing parameters selected by a user from a plurality of selectable parameters *(see Paragraph [0088], "When the user is ready to build the class, the "Generate" button 218 can be selected, for example by clicking with a mouse.")*; and

- displaying a user interface with said plurality of selectable parameters for said custom storage class, said custom storage class specifying a second manner in which said automatic code generator creates source code corresponding to said data referenced by said at least one graphical model, said second manner differing from said first manner *(see Figure 2; Paragraph [0049], "The code generator tool of the preferred embodiment of the present invention is embodied as an application program that presents the user with a Graphical User Interface (GUI) that can be*

used to easily input meta data about an object model."; *Paragraph [0050], "Turning now to*

FIG. 1, by way of an overview, the user of the code generator 100 ... of the present invention

enters meta-data 102 into the code generator 100's GUI."; Paragraph [0083], "Consider now

the entry of the meta data used in connection with the present invention. Such information

representing the meta data is input via the GUI such as illustrated in FIGS. 2-6.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate the teaching of <u>Hunt</u> into the teaching of <u>RTW_UG</u> to include

creating a custom storage class in said modeling and execution environment, said custom storage

class created utilizing parameters selected by a user from a plurality of selectable parameters;

and displaying a user interface with said plurality of selectable parameters for said custom

storage class, said custom storage class specifying a second manner in which said automatic code

generator creates source code corresponding to said data referenced by said at least one graphical

model, said second manner differing from said first manner. The modification would be obvious

because one of ordinary skill in the art would be motivated to apply the same user edits to model

data without introducing errors *(see <u>Hunt</u> – Paragraph [0004]).*

As per **Claim 13**, the rejection of **Claim 12** is incorporated; however, <u>RTW_UG</u> does not

disclose:

- wherein said user-selected parameters for said custom storage class in said user

interface are changed and said source code generated by said automatic code generator is

adjusted to reflect said change in user-selected parameters.

<u>Hunt</u> discloses:

- wherein said user-selected parameters for said custom storage class in said user

interface are changed and said source code generated by said automatic code generator is

adjusted to reflect said change in user-selected parameters *(see Paragraph [0050], "When a*

change to the object model needs to be made, the user can simply update the meta data through

the GUI and regenerate all the files associated with the object model.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to include

wherein said user-selected parameters for said custom storage class in said user interface are

changed and said source code generated by said automatic code generator is adjusted to reflect

said change in user-selected parameters. The modification would be obvious because one of

ordinary skill in the art would be motivated to apply the same user edits to model data without

introducing errors *(see Hunt – Paragraph [0004])*.

As per **Claim 14**, the rejection of **Claim 13** is incorporated; and RTW_UG further

discloses:

- wherein said adjusted source code is displayed in said view of salient aspects of said

source code *(see Chapter 11-6, "MdlStart Function Custom Code dialog box")*.

As per **Claim 15**, the rejection of **Claim 12** is incorporated; and RTW_UG further

discloses:

- wherein said view of salient aspects of said source code includes at least one token,

said at least one token being symbolically representative of a non-displayed segment of code *(see*

Chapter 3-23, "Template makefiles are essentially makefiles in which certain tokens are expanded to create a makefile for your model (model.mk). model.mk is created from the template makefile specified in the Real-Time Workshop page of the Simulation Parameters dialog box by copying each line from the template makefile and expanding the tokens of Table 3-4.").

Claims 16-23 and 26 are computer-readable medium claims corresponding to the method claims above (Claims 1-8 and 11) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1-8 and 11.

9.   **Claims 9, 10, 24, and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over **RTW_UG** in view of **Hunt** as applied to Claims 1 and 16 above, and further in view of **US 2003/0225774 (hereinafter "Davidov")**.

As per **Claim 9**, the rejection of **Claim 1** is incorporated; however, RTW_UG and Hunt do not disclose:

-   wherein said user-selected parameters include a non-portable directive to a compiler.

Davidov discloses:

-   wherein said user-selected parameters include a non-portable directive to a compiler

(see Paragraph [0092], "Command line options or directives for the compiler ..."; Paragraph [0214], "The compiler compiles Java source code produced by the generator according to supplied directives.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate the teaching of Davidov into the teaching of RTW_UG to

include wherein said user-selected parameters include a non-portable directive to a compiler. The

modification would be obvious because one of ordinary skill in the art would be motivated to

conveniently and dynamically create software programs that can be executed on a computer

system.

As per **Claim 10**, the rejection of **Claim 9** is incorporated; however, RTW_UG and Hunt

do not disclose:

- wherein said non-portable directive to a compiler assigns data to at least one memory

location in said electronic device.

Davidov discloses:

- wherein said non-portable directive to a compiler assigns data to at least one memory

location in said electronic device *(see Paragraph [0160], "The data is loaded when the*

application is started, and is saved when the application is destroyed. This type of persistence

uses the device records management system (RMS), for example, non-volatile memory.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate the teaching of Davidov into the teaching of RTW_UG to

include wherein said non-portable directive to a compiler assigns data to at least one memory

location in said electronic device. The modification would be obvious because one of ordinary

skill in the art would be motivated to store data that can be utilized at a later time.

Claim 24 is rejected for the same reason set forth in the rejection of Claim 9.

Claim 25 is rejected for the same reason set forth in the rejection of Claim 10.

Response to Arguments

10.   Applicant's arguments filed on November 10, 2008 have been fully considered, but they are not persuasive.

In the Remarks, Applicant argues:

a)   In Hunt, a class is first created, and then a user is allowed to generate code for the class by specifying settings. That is, the class for which code is generated already exists before the user starts the code generation process outlined in paragraph [0088]. The settings specified by the user in the code generation process are not used to create the class, but merely directs the code generation.

In contrast, claim 1 requires creating a custom storage class by utilizing parameters selected by a user from a plurality of selectable parameters. Hunt merely discusses generating code for a class by using settings specified by a user in the code generation process. However, Hunt does not disclose or suggest creating a class by utilizing any such parameters selected by a user, as required by claim 1.

Examiner's response:

a)   Examiner disagrees. Applicant's arguments are not persuasive for at least the following reasons:

First, with respect to the Applicant's assertion that Hunt does not disclose or suggest creating a class by utilizing any parameters selected by a user, as previously pointed out in the Final Rejection (mailed on 06/09/2008) and further clarified hereinafter, the Examiner respectfully submits that Hunt clearly discloses "creating a custom storage class in said graphical modeling and execution environment utilizing parameters selected by a user from said plurality of selectable parameters" *(see Paragraph [0088], "When the user is ready to build the class, the "Generate" button 218 can be selected, for example by clicking with a mouse.")*. Note that Figures 2-6 of Hunt clearly illustrate a GUI that allows a user to input metadata for a customized class. When the user is ready to build the customized class, the user clicks on the "Generate" button to generate code for the customized class.

Second, the claims recite only "creating a custom storage class" with no further clarification on the claim scope of the term "creating" as intended by the Applicant to cover. Thus, as the claims are interpreted as broadly as their terms reasonably allow (see MPEP § 2111.01(I)), the interpretation of a broad limitation of "creating a custom storage class" as generating code for a customized class and the like by one of ordinary skill in the art is considered to be reasonable by its plain meaning.

Therefore, for at least the reasons set forth above, the rejections made under 35 U.S.C. § 103(a) with respect to Claims 1, 12, and 16 are proper and therefore, maintained.

Conclusion

11.   Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The

Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM.

The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's

supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the

organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding

should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent

Application Information Retrieval (PAIR) system. Status information for published applications

may be obtained from either Private PAIR or Public PAIR. Status information for unpublished

applications is available through Private PAIR only. For more information about the PAIR

system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR

system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Q. C./

Examiner, Art Unit 2191

/Wei Y Zhen/

Supervisory Patent Examiner, Art Unit 2191